AMENDMENTS TO THE SPECIFICATION

Please replace paragraphs **82** *and* **83** *with the following rewritten paragraph:*

[0082] Before this question can be answered, however, there must be some way to define "safe" in a way that the verification engine 250 can use. Note that "safe" here does not mean "error-free" in the sense that correct execution will not lead to undesired results. For example, certain software ~~of Microsoft Corp. is well known to~~ may contain many "bugs," ~~as evidenced by the need to install frequent~~ which, in some cases, may be corrected by installing various "patches." Nonetheless, ~~Microsoft applications (or, indeed, its operating systems) delivered on one or more CDs or through the Internet are~~ such software may be considered to be valid, that is, "safe," in the sense that they consist of a set of instructions and data ~~(however organized into files) created by Microsoft~~ and intentionally delivered for use as they are. Similarly, a user may install code from a trusted source either from a physical medium or through downloading or other network transfer despite having various known or unknown bugs.

[0083] Assume that either the user, a system administrator, or even a trusted remote entity defines some set of pages as being safe. For example, all the executable files of all versions of ~~Microsoft Outlook~~ a particular program could be assumed to be valid, regardless of how error-prone they are likely to be. This is the same as saying that all pages containing the code of ~~these programs~~ the particular program are valid.

Please replace paragraph **87** *with the following rewritten paragraph:*

[0087] It is therefore necessary to have some way to identify a particular page as being "safe." One way to do this would of course be to store a complete copy of the page. This is both wasteful and unnecessary. In the preferred embodiment of the invention, each page assumed to be valid (using any chosen definition) is assigned at least one value that is computed as a function of the entire or partial contents of that page. One example of such a function ~~is any well known~~ may be a hash function, as further explained below. Known invalid pages may be identified using the same procedure.

Please replace paragraph **98** *with the following rewritten paragraph:*

[0098] In the context of binary translation of instructions issued by a virtual machine (see below for further description in connection with Figure 3), any instructions fetched from a translation cache could be assumed to be safe, since their corresponding pages will already have been verified before a previous execution.

Please replace paragraph **104** *with the following rewritten paragraph:*

[0104] Alternatively, if an "execute-unless-blacklisted" policy is chosen, then no white list would be necessary. Such a policy might be useful to block execution of code known to come from an untrusted or undesirable provider. For example, assume that a user (or system administrator) wishes to keep his system clean of code originating from ~~the Microsoft Corporation~~ a particular software publisher. The user could do this by ~~installing a more trusted operating system such as Linux as the operating system 220 and could then include~~ including in the black a list identifiers of known ~~Microsoft~~ code associated with that publisher.